Assignment 6

15-816: Linear Logic
Frank Pfenning

Due
Wednesday, April 18, 2012

This assignment consists of several, somewhat open-ended problems.

> **You should pick one of them, or any of the problems from Assignment 5 that you have not done yet.**

If you have proposed a project in Assignment 5, you should submit a progress report.

You may do these assignments **by yourself or in pairs**. They are somewhat open-ended, so you have to use your judgment as to when you consider the homework completed. Feel free to contact the instructor when you have questions about the extent of a problem.

As usual, you are allowed and encouraged to use *all* resources (papers, lecture notes, technical reports) that you can find, but you must properly cite and acknowledge any resources you use.

Please submit this assignment as a PDF by email. LaTeX templates and macros that may be helpful are available on the course web pages, but you are not required to use them.

Exercise 1 (Parallel Programming and Cost Semantics) Implement some parallel programming language, sample program, and cost semantics as described in the following thesis:

> *Scheduling Deterministic Parallel Programs.*
> Daniel Spoonhower, PhD Thesis, May 2009.
> Available as Technical Report CMU-CS-09-126.

Use Celf for your implementation. The general idea is that execution traces under and SSOS specifications correspond to the computation graphs and heap graphs defined in Chapter 2. Exhibit this relationship through your implementation.

Exercise 2 (Session Types in CLF) Implement the system of session types as discussed early in the course in Celf. Your implementation should be operationally adequate. Express some of the examples, such as the model of the ATM, in your implementation.

Exercise 3 (Parallel Programming) Pick a selection of simple parallel algorithms and either implement them in Celf at a high level of abstraction (such as parallel maximum or parallel bubblesort discussed in lecture), or analyze why they cannot be implemented in this manner. Your main concern should be parallelism in the execution model (and not practicality), but it should be able to run on small examples.

Exercise 4 (Meta-Interpretation) Write an interpreter for a small linear backward-chaining logic programming language in Celf. Start with the highest level of abstraction you can achieve, where resource management and unification are mapped directly to corresponding features of the metalanguage. The rewrite the interpreter at a lower level of abstraction to make some operational aspect such as resource management or unifiction is made explicit.

Exercise 5 (Logical Embeddings) Describe and implement embeddings of classical and intuitionistic first-order logic into linear logic. For these embeddings, first give specifications of appropriate fragments of classical, intutionistic, and linear logic in Celf, and then describe translations on proof terms that preserve provability.

Exercise 6 (Type Isomorphisms) Two types A and B are *isomorphic* if $x{:}A \vdash M : B$ and $y{:}B \vdash N : A$, and M and N compose in both direction to the identity. This is typically considered for natural deduction, but you may also choose the sequent calculus, under an appropriate notion of proof equality.

Explore which types are isomorphic in linear logic. Also give examples of type A and B such that $A \dashv\vdash B$, but A and B are not isomorphic.